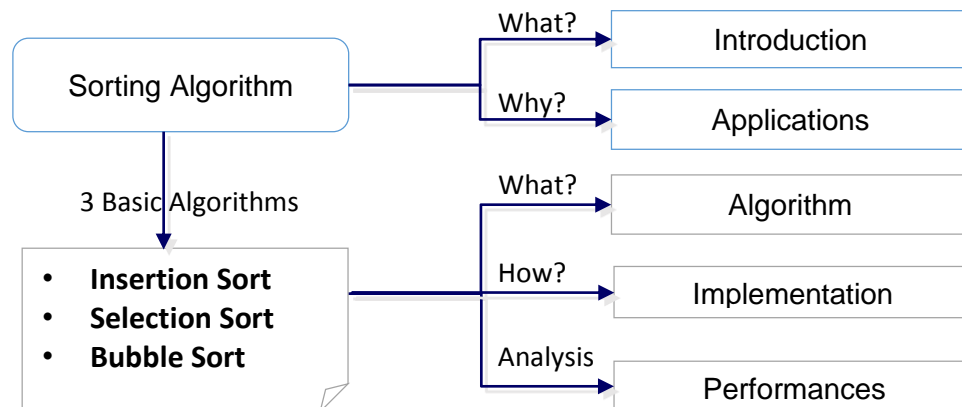


Sorting Algorithms: Insertion, Selection, Bubble and Merge sort

Lesson overview



Merge Sort

1 Sorting Algorithms

1.1 Introduction of Sorting Algorithms

A sorting algorithm is an algorithm that:

- puts elements of an array (a list)
- in a certain order, e.g., ascending numerical order

Question [Link with real world cases]: How do you draw & collate cards during a poker game?

1.2 Applications of Sorting Algorithm

- *Commercial computing*
- *Search for information*
- *Operations research*
-

Question [Link with real world cases]: Sorting applications in our life?

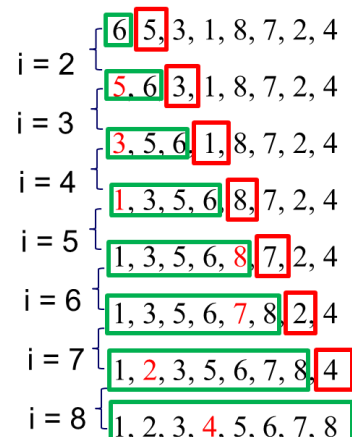
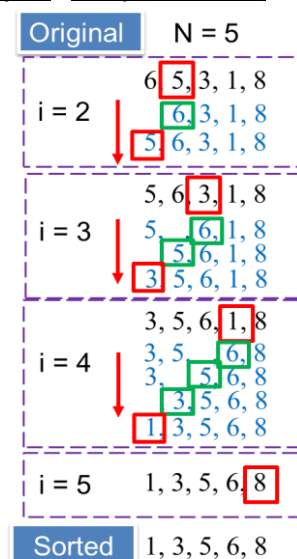
2 Three Basic Algorithms: Insertion, Selection, Bubble

2.1 Three Basic Algorithms: Algorithms

Insertion Sort: maintains a sorted sub-array, and repetitively inserts new elements into it

Example: Array {6,5,3,1,8}

Array {6,5,3,1,8,7,2,4}



Pseudocode // Arr: An Array which is **one-based** **Algorithm: Insertion Sort**

```
N = LENGTH(Arr)
```

FOR i = 2 **TO** N

CurValue = _____

```
WHILE j > 1 AND CurValue < Arr[j-1]
```

```
Arr[j] = Arr[j-1]           // _____
```

ENDWHILE

```
Arr[j] = CurValue // _____
```

ENDFOR

Trace Table Example: Array {5, 4, 8, 1} Algorithm: Insertion Sort

i	CurValue	j	j>1	Arr[j-1]	CurValue <Arr[j-1]	Arr[1]	Arr[2]	Arr[3]	Arr[4]
-	-	-	-	-	-	5	4	8	1
2	4	2	TRUE	5	TRUE		5		
		1	FALSE			4			
3	8	3	TRUE	5	FALSE			(8)	
4	1	4	TRUE	8	TRUE				8
		3	TRUE	5	TRUE			5	
		2	TRUE	4	TRUE		4		
		1	FALSE			1			

Group Learning Activity: *Algorithm: Insertion Sort*

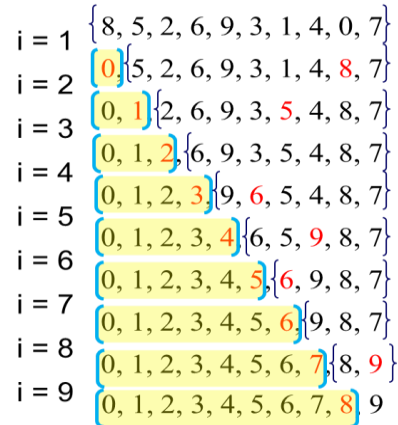
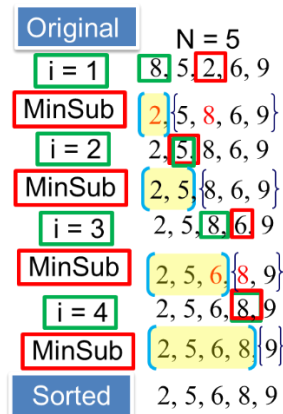
- 1) Complete the pseudocode
- 2) Demo how it works with poker cards: Your Array -- {7, 9, 3, 2}
- 3) Complete the trace table

[illegible]

Selection Sort: repetitively **pick up** the **smallest** element and put it into the **right position**

Example: Array {6,5,3,1,8}

Array {8,5,2,6,9,3,1,4,0,7}



Pseudocode // Arr: An Array which is one-based

Algorithm: Selection Sort

```

N = LENGTH(Arr)
FOR i = 1 TO N-1
    MinSub = i
    FOR j = i + 1 TO N
        IF A[j] < A[MinSub]
            // 
        ENDIF
    ENDFOR
    Temp = A[i] // 
    A[i] = A[MinSub]
    A[MinSub] = Temp
ENDFOR

```

Group Learning Activity:

Algorithm: Selection Sort

- 1) Complete the pseudocode (refer to 2.1: Algorithm: Insertion Sort Trace Table demo)
- 2) Demo how it works with poker cards: Your Array -- {7, 9, 3, 2}
- 3) Complete the trace table

i	MinSub	j	A[j]	A[MinSub]	A[j] < A[MinSub]	Arr[1]	Arr[2]	Arr[3]	Arr[4]
-	-	-	-	-	-	7	9	3	2
1	1	2	9	7	FALSE				

Bubble Sort: repetitively compares adjacent pairs of elements and swaps if necessary

Example: Array {6,5,3,1,8}

Array {6,5,3,1,8,7,2,4}

Original N = 5

i = 1: 6, 5, 3, 1, 8 j = 1
 5, 6, 3, 1, 8 j = 2
 5, 3, 6, 1, 8 j = 3
 5, 3, 1, 6, 8 j = 4
 5, 3, 1, 6, 8

i = 2: 5, 3, 1, 6, 8 j = 1
 3, 5, 1, 6, 8 j = 2
 3, 1, 5, 6, 8 j = 3
 3, 1, 5, 6, 8

i = 3: 1, 3, 5, 6, 8 j = 1
 1, 3, 5, 6, 8 j = 2
 1, 3, 5, 6, 8

i = 4: 1, 3, 5, 6, 8 j = 1
 1, 3, 5, 6, 8

i = 1: 6, 5, 3, 1, 8, 7, 2, 4
 5, 3, 1, 6, 7, 2, 4, 8
 3, 1, 5, 6, 2, 4, 7, 8
 1, 3, 5, 2, 4, 6, 7, 8
 1, 3, 2, 4, 5, 6, 7, 8
 1, 2, 3, 4, 5, 6, 7, 8
 1, 2, 3, 4, 5, 6, 7, 8
 1, 2, 3, 4, 5, 6, 7, 8

Pseudocode // Arr: An Array which is one-based

Algorithm: Bubble Sort

```

N = LENGTH(A)
FOR i = 1 TO N-1
  FOR j = 1 TO N-i
    IF _____
      Temp = A[j] // _____
      A[j] = A[j+1]
      A[j+1] = Temp
    ENDIF
  ENDFOR
ENDFOR

```

Group Learning Activity:

Algorithm: Bubble Sort

- 1) Complete the pseudocode (refer to 2.1: Algorithm: Insertion Sort Trace Table demo)
- 2) Demo how it works with poker cards: Your Array -- {7, 9, 3, 2}
- 3) Complete the trace table

i	j	A[j]	A[j+1]	A[j] > A[j+1]	Arr[1]	Arr[2]	Arr[3]	Arr[4]
-	-	-	-	-	7	9	3	2
1	1	7	9	FALSE				

Group Acting: New group will do group acting of an algorithm based on drawing lots

- ❖ All group members should be involved in the acting
- ❖ Show your creativities and imagination
- ❖ 3 minutes preparation + 2 minutes show for each group * 2

2.2 Three Basic Algorithms: Implementation

Implement the 3 sorting algorithms in Python

- 1) Get the source code from our wiki Platform
- 2) Complete the 3 function for 3 sorting algorithms

2.3 Three Basic Algorithms: Performance Analysis

Computational and space complexity: In terms of the size of the array (N)

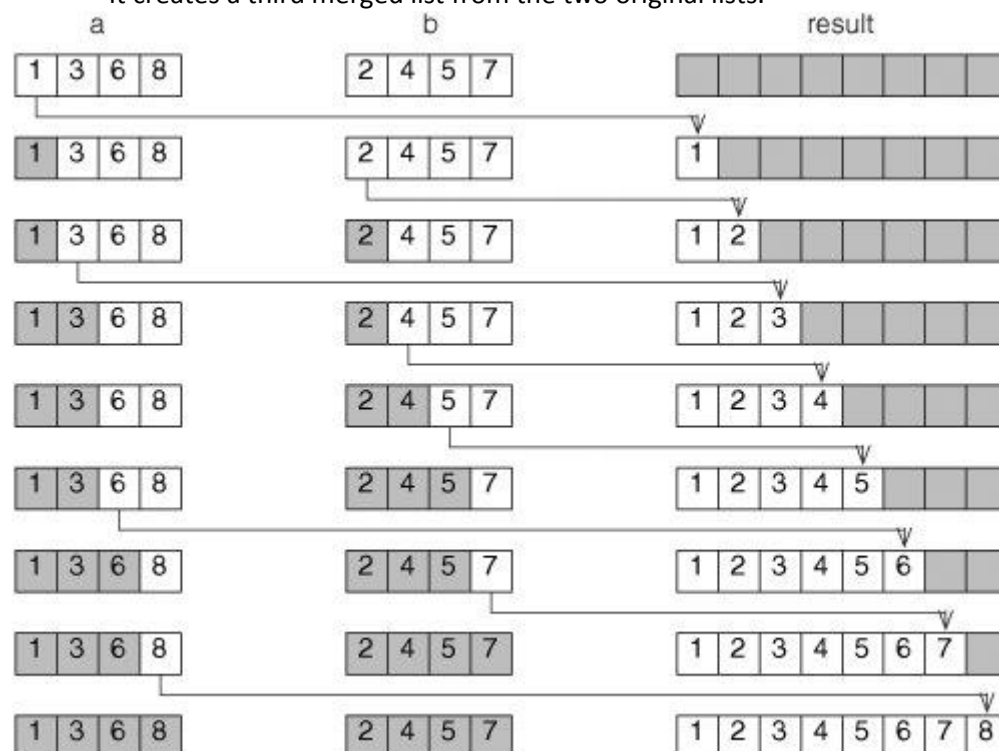
Big O notation - $O(f(N))$

	Computational complexity Time usage		Space complexity Memory usage
	Comparisons	Swaps	
Insertion			
Selection			
Bubble			

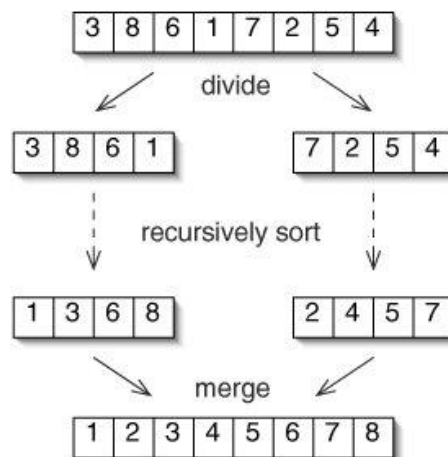
2.4 Extensions: Merge sort

2.4.1 Merging two sorted lists

- The merging algorithm assumes the two lists are in order.
- It creates a third merged list from the two original lists.



2.4 Merge sort: divide-and-conquer

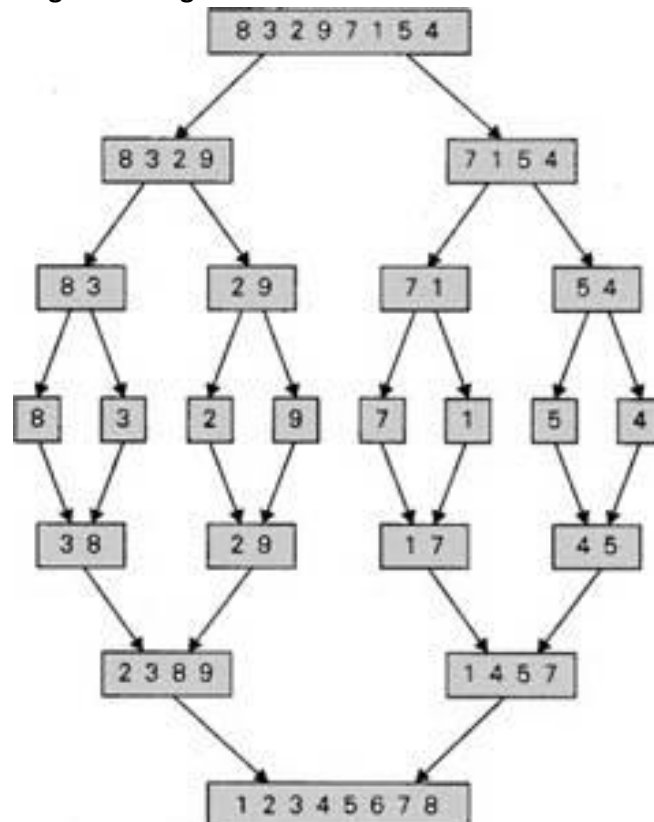


Please complete the following algorithm of merge sort:

```

MERGE_SORT( A, low, high )
  if low < high then
    mid = ( low+high )/2
    MERGE_SORT( A, _____, _____ )
    MERGE_SORT( A, _____, _____ )
    MERGE( A, low, mid, mid+1, high )
  END MERGE_SORT
  
```

An example of merge sort algorithm:



[Quiz]

Given the following list of numbers:

[21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40]

which answer illustrates the list to be sorted after 3 recursive calls to mergesort?

(A) [16, 49, 39, 27, 43, 34, 46, 40]

(B) [21, 1]

(C) [21, 1, 26, 45]

(D) [21]

Given the following list of numbers

[21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40]

which answer illustrates the first two lists to be merged?

(A) [21, 1] and [26, 45]

(B) [[1, 2, 9, 21, 26, 28, 29, 45] and [16, 27, 34, 39, 40, 43, 46, 49]

(C) [21] and [1]

(D) [9] and [16]

[Homework]

- **Complete your handout: exercises**
- **Implement and test each algorithm by yourself, especially the merge sort. Think about recursion.**
- **Practical Exploration: Compare the time complexity of the introduced 4 algorithms, try to plot the running time v.s. Data scale figure using matplotlib**
Explore the algorithm performance based on running the programs →
Plot the running time needed for different array size N
Check the experiment results with theory

